MATHEMATIQ

Der Newsletter der MathSIG
(Interessensgruppe innerhalb der Mensa Österreich)

Ausgabe 17

http://www.hugi.scene.org/adok/mensa/mathsig/

Editorial

Liebe Leserinnen und Leser!

Dies ist die siebzehnte Ausgabe von MATHEMATIQ, dem Newsletter der MathSIG. Die MathSIG wurde gegründet, um die spezifischen Interessen mathematisch hochbegabter Menschen zu fördern. In erster Linie soll sie sich also den Themengebieten Mathematik, Informatik, Physik und Philosophie widmen. Beiträge von Lesern sind herzlich willkommen. Wenn in ihnen mathematische Sonderzeichen vorkommen, bitte ich aber, sie zwecks möglichst einfacher und fehlerfreier Formatierung im TEX-Format einzusenden. Als Vorlage ist eine Fassung des jeweils aktuellen Newsletters im TEX-Format auf Anfrage bei mir erhältlich. Außer Artikeln sind natürlich auch Illustrationen für das Titelblatt willkommen. Die Rechte an diesen müssen aber eindeutig bei euch selbst liegen, Kopieren von Bildern aus dem Internet ist nicht erlaubt.

Hinweis: Autoren sind für den Inhalt ihrer Artikel oder Werke selbst verantwortlich. Die in MATHEMATIQ veröffentlichten Beiträge widerspiegeln ausschließlich die Meinung ihrer Autoren und nicht jene des Vereins Mensa. Die Zusendung von Beiträgen gilt auch als Einverständnis zu deren Veröffentlichung in MATHEMATIQ.

Diese Ausgabe leitet in ein neues Thema (Algorithmik) ein.

In diesem Sinne: Viel Spaß beim Lesen und Lernen!

Claus D. Volko, cdvolko@gmail.com

Der Euklidische Algorithmus

Ein Algorithmus ist im Allgemeinen eine Rechenvorschrift, die einen Rechner anweist, welche Schritte er auszuführen hat, um (irgendwann) zum gesuchten Ergebnis zu bekommen. Ein Beispiel für einen solchen Algorithmus ist der Euklidische Algorithmus zur Berechnung des größten gemeinsamen Teilers zweier natürlicher Zahlen.

Bekanntlich gilt für zwei natürliche Zahlen a und b:

$a = x$ ggT(a, b),
$b = y$ ggT(a, b),

beide Zahlen sind also Vielfaches dieses gemeinsamen Teilers, aber die Faktoren x und y, die ebenfalls natürliche Zahlen sind, dürfen selbst nicht Vielfache dieses gemeinsamen Teilers sein (sonst wäre es nicht der größte gemeinsame Teiler).

Jede natürliche Zahl lässt sich auch auf folgende Art darstellen:

$a = q_1\ r_0 + r_1$.

Solange r_0 eine natürliche Zahl kleiner als a ist, kann man also a durch r_0 dividieren. Man erhält dann einen ganzzahligen Quotienten q_1 und einen Rest r_1. Der Trick ist nun, wiederum r_0 auf diese Weise darzustellen - also als Produkt eines Quotienten q_2 mit dem ehemaligen Rest r_1 plus einem neuen Rest r_2. Wenn man das in die obige Gleichung einsetzt, sieht man, dass die ursprüngliche Zahl a tatsächlich, bis auf ein Vielfaches von r_2 als Rest, ein Vielfaches von r_1 ist:

$a = q_1\ (q_2\ r_1 + r_2) + r_1$
$= q_1\ q_2\ r_1 + q_1\ r_2 + r_1$
$= r_1\ (q_1\ q_2 + 1) + q_1\ r_2$

Der Trick ist nun, das Ganze so lange durchzuiterieren, bis wir zu $r_n = 0$ kommen. Denn dann wissen wir: Es gibt keinen Rest mehr; a ist durch r_{n-1} ohne Rest teilbar. Wir wissen aber auch, dass r_0 ein Vielfaches von r_{n-1} ist. Somit ist r_{n-1} ein gemeinsamer Teiler von a und r_0 - und zwar der größte gemeinsame Teiler. (Warum? Überlegt es euch mal!)

Um den größten gemeinsamen Teiler von a und b zu ermitteln, setzen wir also $b := r_0$ und führen den hier skizzierten Algorithmus aus.

Ein Computer versteht einen in natürlicher Sprache angegebenen Algorithmus nicht. Deswegen werden Algorithmen normalerweise in Computersprachen übersetzt. Nun gibt es in der Programmierung mehrere Paradigmen. Das dominante Paradigma die imperative Programmierung. Dabei weist man den Rechner Schritt für Schritt an, was er zu tun hat. In diesem Fall könnte man zum Beispiel in der Programmiersprache C das Problem wie folgt lösen:

```
int ggt(int a, int b)
{
  while (b)
  {
    int t = b;
    b = a % b;
    a = t;
  }
  return a;
}
```

In der Praxis eines Programmierers wird zwar meistens so oder ähnlich gearbeitet - als eleganter gilt es jedoch, wenn man, wo immer möglich, einen funktionalen Stil pflegt. Die Idee dahinter ist, eine Funktion nur durch andere Funktionen auszudrücken - anders gesagt: im Rumpf der Funktion befindet sich, in C-Syntax, im Prinzip nur eine return-Anweisung. Beim Euklidischen Algorithmus läßt sich das recht leicht bewerkstelligen, weil der Algorithmus rekursiv ausgedrückt werden kann (also als Funktion, die sich mit veränderten Parametern selbst aufruft), etwa wie folgt:

```
int ggt(int a, int b)
{
  return b ? ggt(b, a % b) : a;
}
```

Es ist schon eine Kunst, einen möglichst eleganten und effizienten Algorithmus zu finden, der ein gegebenes Problem löst. Da es sich dabei um eines meiner Spezialgebiete im Studium handelte, werde ich wahrscheinlich noch öfter hier in MATHEMATIQ über dieses Thema schreiben.

Claus D. Volko, cdvolko@gmail.com